ECS 152A: Computer Networks Project 3

Mitmproxy

Professor Zubair Shafiq

Krystal Chau, SID: 920918540 Jacob Feenstra, SID: 921423591

November 3, 2023

1 File Submissions

1. `proj3.py`
2. `proj3-ChatGPT.py`
3. `wireshark-response1.png`
4. `wireshark-response2.png`
5. `mitmproxy-response-headers.png`
6. `mitmproxy-flow.png`

2 Questions

1. Wireshark

   (a) Can you tell what the secret key is?

   From what Wireshark provides us, we cannot tell what the secret key is. We took the liberty of running a capture of the traffic against a python script (using `dpkt` and `requests`). We were able to distinguish the packets sent to and from the prompt URL as two `DNS` protocol packets. We used some `dpkt` code to see if we could identify anything on the packet-level, but we were only able to print the URL that is being queried. `requests` gives us the ability to print-out all of the headers in the `HTTP GET` request. No luck there either. See our Wireshark screenshots below, for each DNS packet.

Figure 1: DNS Packet 1

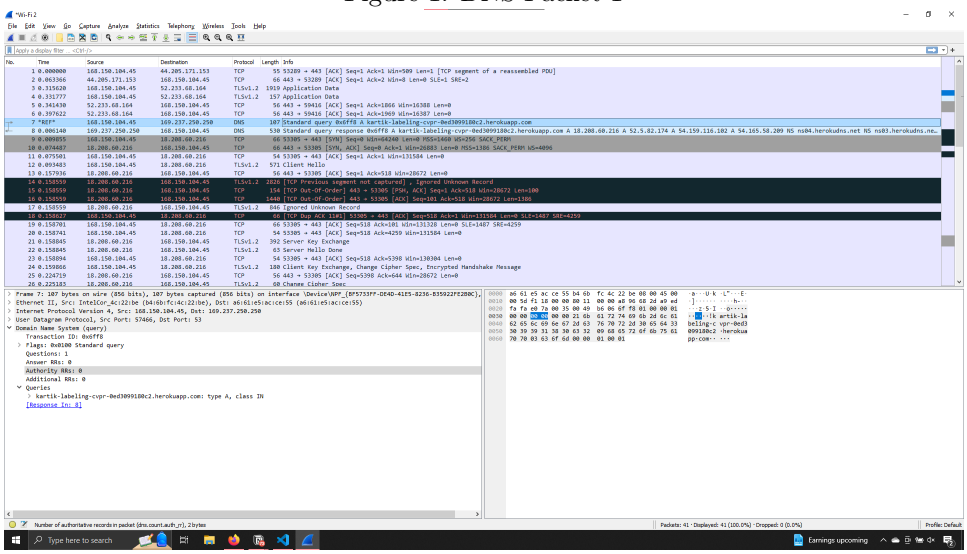
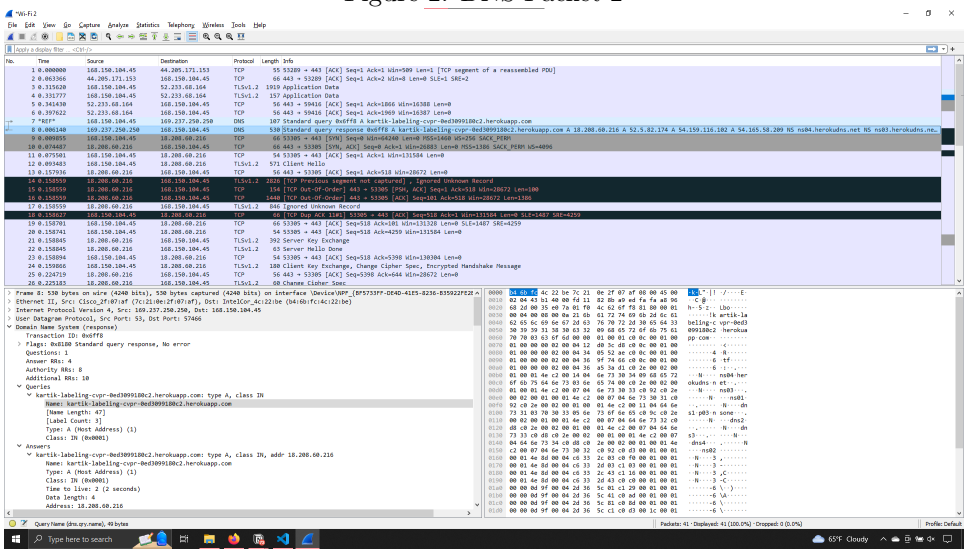


Figure 2: DNS Packet 2



   (b) If yes, what is it? If not, why so?

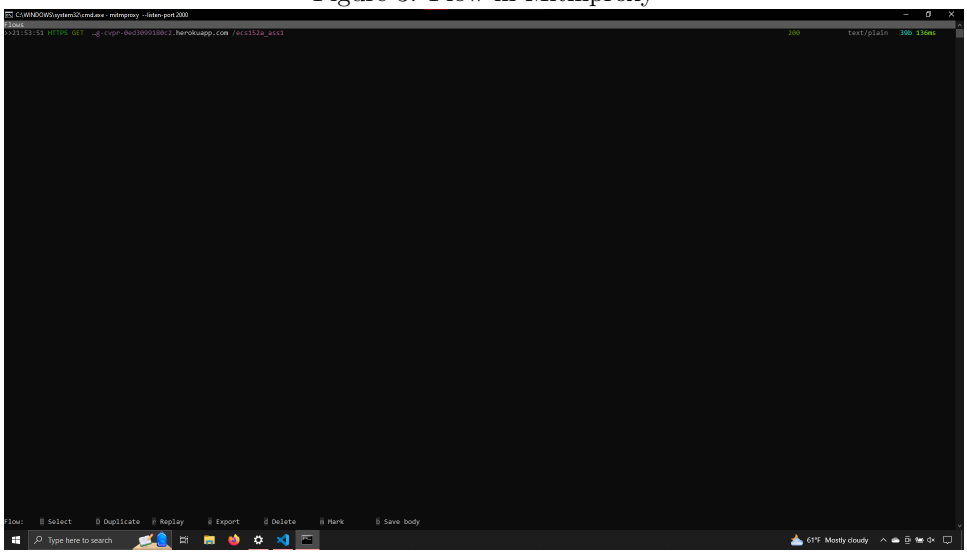
   To our knowledge, DNS is not encrypted in transit but encrypted end-to-end; we are probably unable to find the secret because it is hidden behind the encryption (and Wireshark's captures likely display the arrived packet, after transmission).

2. Mitmproxy

(a) Can you tell what the secret key is?

Yes! We see one of the response headers from the server is related to our class, and is in a key format. The response packet in question is HTTPS, and can be seen below:

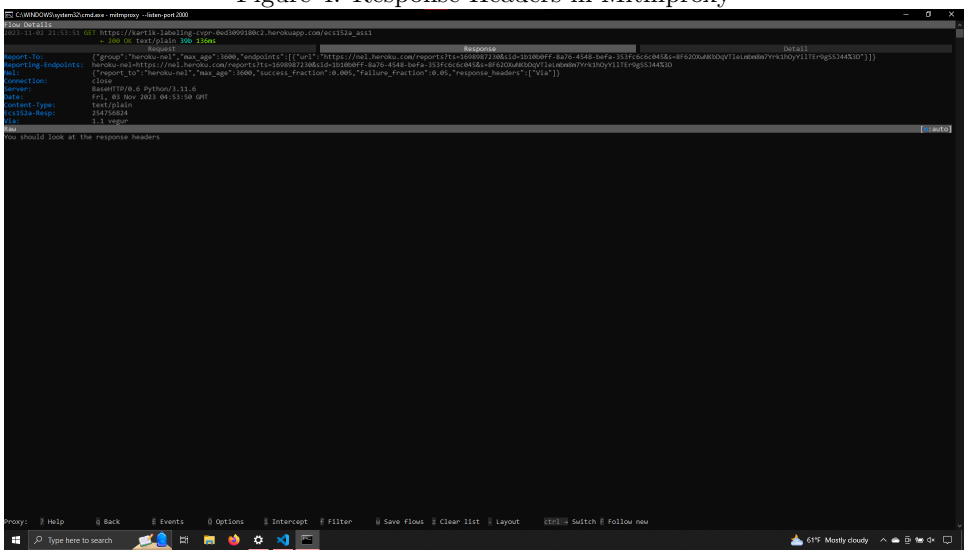
Figure 3: Flow in Mitmproxy



(b) If yes, what is it? If not, why so?

The secret key is the value for the `Ecs152a-response:` response header, `254756824`. We believe it is this because the header name is related to the class, and if we do not provide a student ID, the header instead returns a value indicating that it needs a student ID.

Figure 4: Response Headers in Mitmproxy



(c) If your answer to 2 changed from above, what do you think Mitmproxy changed?

The key (no pun intended) lies in deciphering the acronym in Mitmproxy: Man in the Middle. By inserting Mitmproxy in the transmission between client and server, we can "inspect" the contents of the travelling packets. Wireshark has no such capability; a Wireshark capture will simply show the packets that are received by the client, or received by the server. They do not show packets during "live" traffic, so to speak. They are encrypted upon arrival. They are, however, not encrypted during transit. Mitmproxy leverages this, scanning and outputting the packet contents during their routing. So where we could not see the response (or even request) headers in Wireshark, we are able to do so in Mitmproxy, and thus inspect our key. The secret key is there in Wireshark... we just can't decrypt it.

3 References

Chat GPT-3.5 Session Link Sending a Request
OpenAI. (2023). ChatGPT [Large language model]. https://chat.openai.com

We used a code sample in the `request` library docs to figure out the syntax for sending out an HTTP GET, and also found a method to print out the header of the response (`{REQUEST_OBJECT}.headers`). This was mostly to play around with the library; the print method is not involved with the logic of Part 3 in any way. We also referenced an example of `dpkt` DNS packet parsing, linked below. It inspired our `printout()` function, and some of the DNS operations needed to analyze the packets from the Wireshark capture. This did not play any role in Part 3's requirements either.

print_dns_truncated.py

Fun assignment! Really enjoyed it Hari (or whoever is reading this). :)